Practice Midterm 1 Solutions

This page contains solutions to Practice Midterm 1.

Q1) C++ fundamentals

```
bool findFreeBlock(Grid<string>& seatGrid, int k, GridLocation& loc) {
    for (int r = 0; r < seatGrid.numRows(); r++) {
        int count = 0; // count of num consecutive empty seats in current
row
        for (int c = 0; c < seatGrid.numCols(); c++) {
            if (seatGrid[r][c].empty()) {
                count++;
                if (count == k) {
                    loc = { r, c - k + 1 };
                    return true;
                }
            } else {
                count = 0;
            }
        }
    }
    return false;
}
```

Q2) ADTs

```
int reseatGroup(Grid<string>& seatGrid, Map<string, Set<GridLocation>>&
reservationDB, string groupCode) {
    // Start by "unlocking" the seat assignments from existing reservation
    Set<GridLocation> oldSeats = reservationDB[groupCode];
    int k = oldSeats.size();
    for (GridLocation seat : oldSeats) {
        seatGrid[seat] = "";
    }

    GridLocation loc;
    if (findFreeBlock(seatGrid, k, loc)) {
        // Update reservationDB with new block of seats
        reservationDB[groupCode] = getSeatsForBlock(loc, k);
    }

    // mark seat assignments (will restore old or set new)
    for (GridLocation seat : reservationDB[groupCode]) {
        seatGrid[seat] = groupCode;
    }

    // Take a set difference between seatsNow and oldSeats.
    // Elements that are in both will be removed.
    Set<GridLocation> changedSeats =
oldSeats.difference(reservationDB[groupCode]);
    return changedSeats.size();
}
```

Q3) Code study: ADTs and Big-O

a) $O(N^2)$

b)

```
Input vector   Expected result    Actual result
-----------------------------------------------
 {4, 4, 5}           {5}            {4, 5}
```

c) O(N)

d)

```
Input queue     Expected result    Actual result
-----------------------------------------------
 {4, 1, 4}           {1}            {1}
```

e) None

f)

```
Input stack   Expected result    Actual result
-----------------------------------------------
{7, 4, 4}          {7}               {7}
```

g)

```
Input stack   Expected result    Actual result
-----------------------------------------------
{4, 7, 4}           {7}           {4, 7}
```

Q4) Recursive fractal

```
double drawBoxTrio(GPoint upperLeft, double length, int order) {
    if (order == 0) {
        return drawYellowBox(upperLeft, length);
    } else {
        double third = length / 3;
        drawBoxTrio({upperLeft.x + 2*third,  upperLeft.y}, third, order -
1);
        drawBoxTrio({upperLeft.x,  2*third + upperLeft.y}, third, order -
1);
        return drawBoxTrio(upperLeft, 2*third, order - 1);
    }
}
```

Q5) Recursive backtracking

```
int countAlphabeticWords(Lexicon& lex, string sofar) {
    int count = 0;

    if (!lex.containsPrefix(sofar)) {
        return 0;
    } else if (lex.contains(sofar)) {
        count++;
    }
    char start = 'a';
    if (!sofar.empty()) {
        start = sofar[sofar.length()-1];
    }
    for (char ch = start; ch <= 'z'; ch++) {
        count += countAlphabeticWords(lex, sofar + ch);
    }
    return count;
}

int countAlphabeticWords(Lexicon& lex) {
    return countAlphabeticWords(lex, "");
}
```